Lehigh University
Lehigh Preserve

Theses and Dissertations

1967

A single-machine scheduling problem in the box container industry

Waldemar Saure
Lehigh University

Follow this and additional works at: https://preserve.lehigh.edu/etd

 Part of the Programming Languages and Compilers Commons

Recommended Citation
Saure, Waldemar, "A single-machine scheduling problem in the box container industry" (1967). *Theses and Dissertations*. 3550.
https://preserve.lehigh.edu/etd/3550

This Thesis is brought to you for free and open access by Lehigh Preserve. It has been accepted for inclusion in Theses and Dissertations by an authorized administrator of Lehigh Preserve. For more information, please contact preserve@lehigh.edu.

A SINGLE-MACHINE SCHEDULING PROBLEM IN THE BOX CONTAINER INDUSTRY

by

Waldemar Saure

A Thesis

Presented to the Graduate Faculty

of Lehigh University

In Candidacy for the Degree of

Master of Science

Lehigh University

1967

CERTIFICATE OF APPROVAL

This thesis is accepted and approved in partial fulfillment of the requirements for the degree of Master of Science.

August 9, 1967
Date

Gary E Whitehouse
~~Prof~~essor in Charge

Clifton F. Gould
Head of the Department

ACKNOWLEDGEMENTS

The author wishes to express his sincere gratitude to Professor Gary E. Whitehouse for his time and guidance given in support of this thesis. Also Profs. George E. Kane and Wallace J. Richardson who were members of my advisory committee.

Special mention should be made of two fellow students: Lawrence E. White, with whom I had discussions throughtout the course of this project, which were stimulating and enjoyable, and Samares Kar who patiently read the draft of this report and improved on English.

I am thanking the members of the Company, where I found this problem, for their interest, cooperation and financial support.

TABLE OF CONTENTS

ABSTRACT

Waldemar Saure, M.S., Lehigh University, October 1967.

A Single-Machine Scheduling Problem in the Box Container Industry.

The single-machine scheduling problem for a box container plant is defined. The problem is to assign orders to a printing press, which can be formulated in a linear programming framework. The resulting matrix, however, is too large for any realistic set of data to be computed in a reasonable time on a computer.

Hence the approach taken is to use suboptimizing algorithms. Two algorithms are devised and tested with simulated data. An efficiency quotient is defined to evaluate the resulting production schedules against an ideal, normally not attainable schedule. Some of the generated schedules are uneconomical in the sense that high production cost would be incurred if implemented. This is due in part to a constraint placed upon them, which requires all orders to be filled exactly. In actual practice production scheduling is more flexible, i.e., orders can be over or under shipped.

The efficiency quotient of the production schedules developed by the two algorithms averages around 0.97. This result gives cause for optimism, since the underlying view of this study is that future production scheduling systems in the industry will be computerized systems.

Chapter I

INTRODUCTION

The topic of this paper deals with a problem of the container industry. This industry supplies dairies, juice producers and other liquid food stuff industries with containers in which their products are delivered. One of the major problems faced by a container plant is that of scheduling its printing presses. These machines colorpress in a continuous process a group of various designs on paper board. Paper board is fed into the press from stock rolls and the printed cardboard (the colors are instantly dry) is wound on reels for further processing.

In order to make maximum use of the printing press and the subsequent processing machines, the plant prefers to lay out as many cartons across the paper strip as possible. Cardboard is kept in inventory in different widths. As mentioned, the printing process is continuous, but is interrupted when the scheduled quantity for boxes of particular specifications has been printed. A so-called "make ready" time is then encountered to reset the machine according to the production plan for a different set of orders. This plan is prepared by the production scheduler. His problem may be put in the following terms: how should a certain number of carton-designs be printed on card-board having a specified calipher, discrete width and limitless length in a way such as to accomplish one or more of the following goals:

1. minimize paper wastage,

2. maximize rate of production of this part of the manufacturing process,

3. minimize transition and printing plate costs,

while taking into consideration certain constraints (e.g. due dates for shipment, physical characteristics of the available machinery).

Orientation of this study

The author has had an occasion to study the behavior of a production schedule in a box plant east of the Mississippi River.[1] From this study some knowledge was gained about decision rules employed by the human being to solve the problem concerned. With this knowledge serving as a key to comprehend the intrigues of the problem, algorithms will be developed to schedule printing presses. The algorithms will then be programmed in a common computer language (Fortran IV).

The idea is not to reproduce the human behavior but to devise models to automate production scheduling. It is intended to test schedule as arrived at by employing the programmed decision rules against an ideal, usually not attainable, production schedule. The merits of the approaches taken will be rated on economic grounds.

The study tries to solve an actual problem in a specific area. The motivation is to suggest models which in conjunction with computors can be applied to production scheduling,

---

[1] To protect the anonymity of the Company.

since with the development of electronic computing devices more and more integrated man-machine systems have been devised. This thesis will provide another step to apply these systems into the field of decision making.

Other work in the area

Developments of quantitative anylysis have found wide acceptance in the paper making and paper converting industry. As a survey indicates (7) optimizing techniques such as linear programming have been extensively applied to the paper trim problem (8). Other areas of application include pulpwood procurement, machine loading, product mix determination and the scheduling of paper mills (4). Words such as control-variable, sub-optimizations, linear inequality, Monte Carlo technique are in the vocabulary of industry men active in work to make the paper industry more efficient.

The problem studied in this paper seem to be one of a large class of scheduling problems which are characterized by a sequence of operations to be performed. The job shop sequencing problem (1) and the assembly line balancing problem (1o) are also basically of combinations in nature.

An interesting approach to a scheduling problem in the corrugated container industry is offered by reference (11). The author of this paper developed programs to schedule corrugators based on heuristic problem solving techniques.

Chapter II

BACKGROUND AND THE CURRENT TREATMENT OF THIS PROBLEM

Paper board, the major ingredient for box containers, is received from paper mills. It is kept as an inventory item in the container manufacturing plant in large rolls. Since for different box sizes and designs various grades of paper are needed, reels stored vary in weight from about 10 pounds per 1,000 square feet - for the tissue-paper - up to well over 200 pounds per 1,000 square feet - for the heavy card-board type paper.

Ordered card boxes are specified among other things by the type of printing the customer desires. Customers sometimes furnish the printing plates, but usually subcontractors supply them to the container industry. With the appropriate colors at hand a printing press could then start to operate. Printing is an uninterrupted process. At one end of the printing press, the paper board is unwound from a reel, the printing is done in several stages and then the cardboard is wound up at the other end of the machine in large rolls. Storing of these rolls serves as a buffer effect, to absorb the temporary differences in the processing time of printing, and extruding - the next performed operation.

The extruder applies a coat of polyethylene to both sides of the printed cardboard. A deckling operation can also be performed at this work station, to increase the strength of the cardboard and herewith the strength of the finished

container. In a deckling operation a tissue paper is combined with the printed cardboard. Immediately after the polyethylene has been applied, a tissue paper strip is pressed against the unprinted side of the already coated cardboard. The polyethylene glues both paper strips into a homogeneous unit. A third polyethylene layer is then applied on the tissue paper side, which later will be the inside of the box container. Extruding is also a continuous process. At one side of the extruder the printed cardboard is unwound from the reels, and if deckling is specified, a tissue paper is unwound from reels. The polyethylene coats are applied and at the end the cardboard is wound in large reels. Storage of these reels has a smoothing effect on the production line between extruder and the latest stage in this line - the cutter.

In the cutting operation, the shape of the unfolded box is cut from the printed and folded cardboard. The cardboard is fed into the cutter and a reciprocating die-set does the cutting and creasing. At the end of the machine, workmen inspect for rejects and put the cartons on pallets which are kept in inventory before shipment to the customer.

The current solution of this problem

In this section we will describe how this problem is being currently treated by the production scheduler. Since we are dealing with an actual situation, some technical terms of the container industry will be used, because this will make it easier to state the problem (Appendix A contains a glossary

defining or explaining some of the more frequently used terms). Inputs to the production scheduler, orders which are received by the box container factory specify the following things:

1. the quantity of boxes to be delivered,
2. the size of the box,
3. the type of the box (numerous box design exist for boxes of a specified size),
4. design-number,
5. the color(s) of the printing,
6. the delivery date.

All incoming orders are piled up over a certain time period. Order in this context one item on an order sheet, having the attributes as defined above. So it might happen, and in fact it is a frequent occurance that many orders are combined on an "order sheet" (from one customer). The production scheduler selects orders, having a delivery date in the week past the production period, from the incoming ones as well as from the file "permanent order sheets". Since scheduling is done in weekly intervals, in the week before scheduling starts, it is obvious that there must be about three weeks lead time for the orders.

Having collected all the orders with the proper shipping dates, the production scheduler then sorts the orders according to "compatibility". This is a screening procedure which puts feasible combinations of orders in one pile (only specific combinations of type and size are feasible).

From this point on, one of the several piles is considered at a time and the production scheduler will apply different criteria to arrive at the proper schedules. In order to understand this let us look at a specific pile that has two container capacities. The physical limitations of interest for the printing press are as follows: there are three "streams" arranged across the machine, utilizing the maximum possible width. Two streams being equipped to process "capacity A" type containers, the remaining stream takes care of "capacity B" type containers. Each stream has two "rows", and in each stream up to three distinct colors can be printed simultaneously. To begin with, each order of capacity A can be processed on any of the four rows, which will be designated for the sake of brevity as S1:R1, R2 and S2:R1, R2. Capacity B type containers can be processed in one of the rows named as S3:R1, R2.

The production scheduler then selects the largest order from each container capacity. The two orders are then arbitrarily assigned, say to S1:R1 (container capacity A) and S3:R1 (container capacity B).

The next step usually taken is to select the next largest order based on "color compatibility" with the assigned orders and which therefore can be processed in the other four rows still not occupied. The scheduler's pattern of activity is not exactly rigid. He will generally select the longer orders, first checking on color feasibility. Once all six rows have an assignment of orders, practically speaking the machine

could start printing, after the proper setting and adjustments had been done, utilizing the full width of the press. Numbering consecutively and starting with run no.1, the press would be stopped if the smallest of the orders is already printed. Having this picture in our mind, we again follow the scheduler, who schedules for the present empty row another feasible assignment. Run no.2 can start and so forth.

The guiding principles of the scheduler's work are color compatibility as mentioned, and make ready considerations. Between two runs the press needs a reset of printing plates and possible color changes in a row. The time for this make ready is dependent on the number of color and plate changes but <u>not</u> proportional to it. That is to say to stop the machine and make four plate changes and two color changes does not take twice as long as for two plate changes and one color change. The human scheduler makes use of results from time studies at hand and tries to avoid color changes as much as possible. A production schedule is considered completed if all orders have been assigned from one pile.

The most costly aspect of an imperfect but still feasible schedule is the "run out". The term is defined as follows: "The procedure of printing in less than the full number of rows is called a run out (paper is fed across the full width of the machine, but on a fraction of this printing takes place)". Before such a run out operation is to be undertaken the file permanent order sheet is examined for fitting

orders, which after completion would be kept in inventory until the shipment is due. Another alternative is the "splitting" of orders. In this case one order will run in more than one row. Additional expenses are incurred if they run simultaneously since printing plates for each row must be available. Another solution is to process less than the ordered quantity and supply to the customer less than he ordered, to obtain a production schedule meeting more economically the objectives described in the last chapter.

As a matter of fact, an effort may be made to procure some orders from some well known customers to avoid a run out. It must be remembered that unassigned rows cause a high percentage of "scrap paper" and lowers the "machine efficiency". As an analysis of the raw material cost shows, the paper board is the overwhelming single cost item. It can therefore be understood that all feasible means are employed to weed out production schedules of this kind.

The foregoing description of the work of a production scheduler is based on observations of the author. A number of considerations have been omitted as any experienced scheduler will note. For example the revision of schedules due to "hot order" (a customer calls in at 9 a.m. to have some cartons printed and ready for shipment at 4 p.m.). Also, the author might not have noticed all the criteria for decision making by the human scheduler to come up with his production plan. It is nevertheless hoped that the main principles of and considerations for the scheduler's work have been stated.

Chapter III

MATHEMATICAL ALGORITHMS FOR THE SOLUTION OF THE SCHEDULING PROBLEM

Formal statement of the problem

There are a set of orders (O_1, O_2, ..., O_n), for boxes of a certain type and size. The distinct attributes associated with each order are the following:

1. the number of cartons to be scheduled,
2. a number specifying the printing pattern,
3. the number of colors involved in printing (one or two).

The printing is done on a printing press having streams arranged across the width of the machine. In any one of the streams up to three distinct colors can be printed. There are either one or two rows in each of the m streams.

The problem now arises to prepare a feasible production schedule specifying in which sequence to print the cartons, in other words which order will be printed in which stream (and in which row, if for the set of orders two rows are arranged in one stream), and in what sequence.

A feasible production schedule is defined as follows: it must take into account all orders of the set. If there are two rows in a stream, the two orders printed simultaneously in one stream must be color-compatible.

An optimal schedule is a feasible schedule having the

minimum longest run, i.e. the least consumption of paper board, with the minimum possible color changes.

Nature of the scheduling problem

We will concern ourselves here with a simplified version of the earlier stated scheduling problem. Suppose there are n orders on hand, to be assigned for a printing press having m streams. Each order can be placed on any of the streams. Let us assume one row per stream so there is no color interference problem.

The objective will be stated as follows: sequence the n orders in the m streams in such a manner, so that there is a minimum longest run.

This objective differs from the one stated in "The formal statement of the problem" in the following manner:

1. the problem of color compatibility is neglected,

2. the number of color changes is of no concern, i.e. it is not tried to minimize the number of color changes.

So the following mathematical formulations are applicable only to printing press set ups having only one row per stream and for sets of orders of one container size.

Linear programming formulation

To cast the machine scheduling problem as a linear programming problem it may be described as follows:

$$\text{MIN} \sum_i \sum_j \sum_k C_k * X_{ijk}$$

subject to $$\sum_j \sum_k X_{ijk} = \frac{N_i}{k} \quad \text{for each } i$$

$$\sum_i X_{ijk} = 1 \quad \text{for all } j \text{ and } k$$

and the nonnegativity constraint $X_{ijk} = 0$;

where X_{ijk} is one (zero otherwise) if the assignment of the i^{th} order extends into, or beyond the k^{th} interval of the j^{th} stream.

C_k "cost factor" associated with X assigned to the k^{th} location.

$i = 1, 2, \ldots, n$ (n = number of orders)

$j = 1, 2, \ldots, m$ (m = number of streams)

$k = 1, 2, \ldots\ldots$ (index of "cost factor")

N_i = quantity to be scheduled for the i^{th} order.

An illustrating example

Suppose the following orders should be scheduled,

Order	$i =$	1	2	3	
Quantity	$N_i =$	10M	12M	18M	$(M = 10^3)$.

We have $j = 1, 2$ (two streams)

and define $L_1 = \min_i (N_i) = 10M$

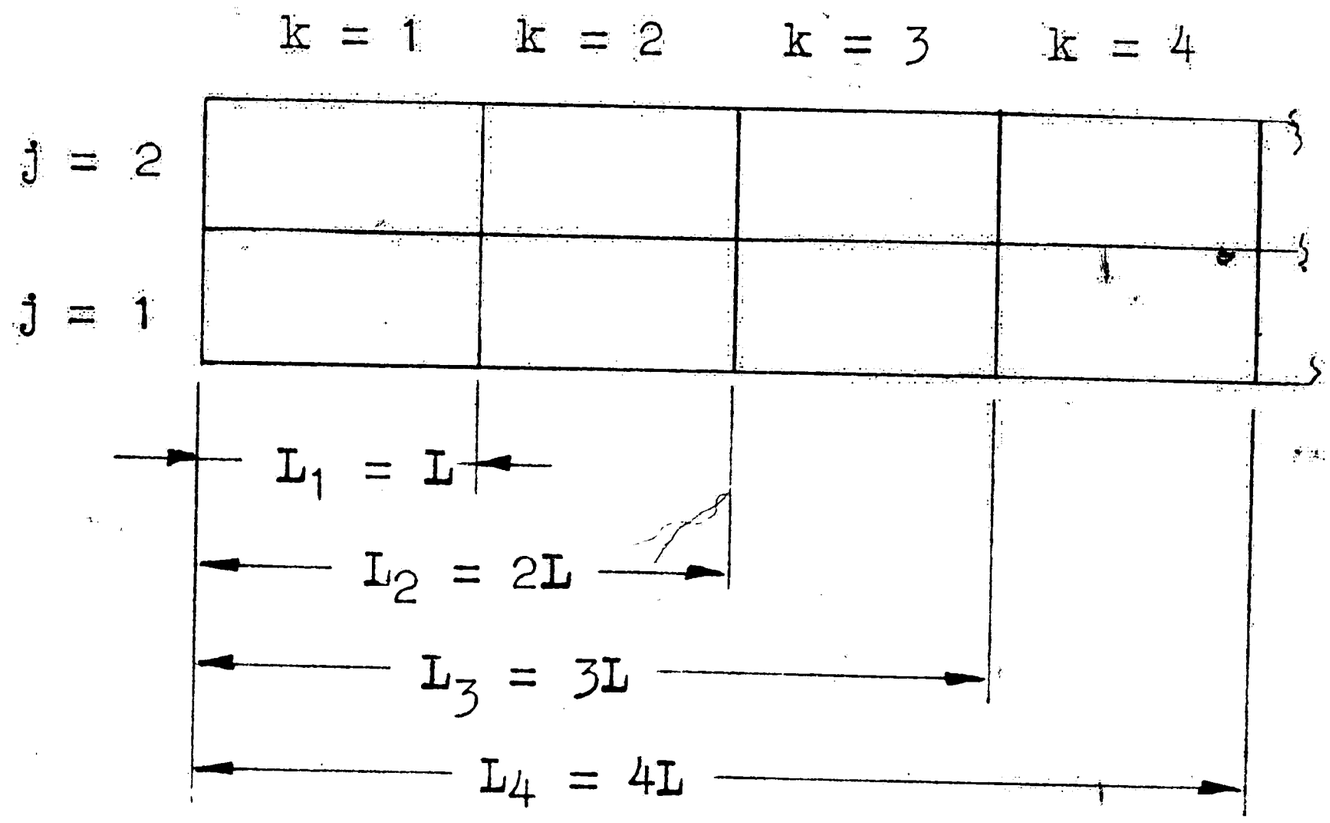
and maximum $L_{max} = \sum_i N_i = 40M$

the "cost factor" index becomes

$$\frac{L_{max}}{L_1} = \frac{40}{10} = 4 \quad \text{(rounded to an integer)}$$

$k = 1, 2, 3, 4$

We show graphically as follows:



The "cost factors" C_1, C_2, C_3, C_4 have to be of increasing magnitude,

"cost factor" = f (C_k) increasing function with k.

We assign the following values to the "cost factors":

$$c_1 = 1.0$$

$$c_2 = 2.0$$

$$c_3 = 3.0$$

$$c_4 = 4.0.$$

Adding slack variables and artificial variables, the problem looks as follows (see Table 1).

The objective coefficients of the slack variables have zero value, the artificial variables have a very large value (bigger than everything they are ever compared with) in the objective function.

Exhaustive search as a solution

It could be attempted to generate all possible distinct production schedules and select an optimal one. The resulting computational work involved would be as follows: having m streams in which to sequence n orders, we make the simplifying assumption that in each stream $\frac{n}{m}^2$ orders are arranged (this would be a feasible schedule, utilizing the least amount of paper board, if all orders had the same quantity specified). Now in each stream there are $\left(\frac{n}{m}\right)$ possible ways to sequence these orders, or combined there are $m * \left(\frac{n}{m}\right)^3$ possible ways to schedule these orders.

---

[2] Rounded to nearest integer.

[3] Assuming the jobs are preassigned to the m rows.

TABLE 1

	i = 1								i = 2								i = 3																								
	j = 1				j = 2				j = 1				j = 2				j = 1				j = 2																				
	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4	0	0	0	M	M	M	0	0	0	0	0	0	0	0	0	
i = 1	1	1	1	1	1	1	1	1																	-1			1												1.0
i = 2									1	1	1	1	1	1	1	1										-1			1											1.2
i = 3																	1	1	1	1	1	1	1	1			-1			1										1.8
	1								1								1														1									1
		1								1								1														1								1
			1								1								1														1							1
				1								1								1														1						1
					1								1								1														1					1
						1								1								1														1				1
							1								1								1														1			1
								1								1								1														1		1

NOTE: Blank spaces in matrix are zeros.

Introduction to suboptimizing approaches to the scheduling problem

In the next two sections two essentially different algorithms for the scheduling problem will be described. The first algorithm has a straightforward structure and converges very fast. The second algorithm begins with choosing a solution to the scheduling problem, indicating one possible, possibly poor way to process the desired number of orders. The algorithm then improves this production schedule by substitution and an exchange of orders.

The next chapter reports the results, that these algorithms produce when confronted with different sets of data. The simple examples in the following sections, are created to illustrate the different algorithms. For the empirical solution in the next chapter a series of simulated examples will be used.

Largest - quantity - first algorithm

The linear programming formulation, presented in the previous chapter, of the scheduling problem is one way of looking at this problem. However, it was not a good way to solve the problem because of the large number of variables and constraints involved.

Making the same assumption on which the linear programming formulation was based, we will now describe a method that will provide rigid rules of how to sequence orders in order to arrive at a production schedule. The procedure is the result of a conscious attempt to avoid large run outs in a printing operation (recall that a run out is the most costly aspect of

of an imperfect production schedule).

It is called largest-quantity-first algorithm because of the principle to assign the currently largest order in quantity of the list to the production schedule. The properties of this algorithm include a fast assignment of all orders and a conceptually straightforward structure.

The largest-quantity-first algorithm begins with an assignment of the first order in the list (vector) of orders to be assigned, to any one, say the first of the m streams. The vector describing the orders was first sorted according to quantity, that is to say all orders were arranged quantitywise in a descending sequence. The remaining m-1 streams are filled by assigning always the currently first order of the sorted list to the schedule.

The next step is to select the stream having the shortest run. To this stream the first order of the remaining list is assigned. The steps are then repeated:

1. select row having the shortest run,
2. assign first order of the existing list to the selected row.

Once all orders of the list have been assigned the production schedule is completed.

A short example of this algorithm is given below:

Consider the following set of orders

Order No.	1	2	3	4	5	6	7	8	9
Quantity required	30M	95M	160M	55M	120M	240M	80M	70M	110M

assume m = 3 streams; $(M = 10^3)$

the requirement vector sorted according to quantity would appear as follows:

Order No. 6 - 3 - 5 - 9 - 2 - 7 - 8 - 4 - 1.

The algorithm, after randomly chosen assignments of the first three orders in Stream 1, 2, 3 respectively, leads to the following production schedule.

Stream 1	Order No.	6	8		
	Quantity	240M	70M		
Stream 2	Order No.	3	2	4	
	Quantity	160M	95M	55M	
Stream 3	Order No.	5	9	7	1
	Quantity	120M	110M	80M	30M

To make the last assignment (30M) a "tie" has been arbitrarily resolved between Strean 1 and Stream 3; both having an accumulated quantity assigned of 310M before the final assignment.

Color-selection algorithm

The second algorithm developed to solve the scheduling problem begins with some feasible production schedule containing all the orders. The algorithm, after deciding on a reasonable starting point, proceeds by substitution of orders in the sequence. If a substitution is possible another one will be tried and so forth. Finally after all feasible substitutions have been made, the algorithm terminates having also attempted an exchange of orders.

The list (vector) with the total number of orders to be

scheduled is sorted on the basis of color requirements as follows: there will be m-1 subvectors containing orders with the first, second, ..., m-1 most frequent colors (m = number of streams on the printing press). The last subvector takes care of all orders in the list that are not contained in the first, second, ..., m-1 subvector. For ease of computer programming a color is identified only by the first of the three digits used by the container industry for a color. So there is a slim chance that a subvector actually contains more than one color. It will later be seen that this has no impact on the final production schedule.

The elements (orders) of the subvectors are then assigned to the first, second, ... and (m-1) the stream respectively. This constitutes a possible production schedule. The objective now is to improve this initial production schedule i.e. eliminate, as much as possible, run out, and in turn production costs.

The first substitution attempt is made in the stream having the shortest run. This could be called "packing". The shortest order of the longest run is added to the shortest run. If the new sum in this stream is less than the sum of the longest run, substitution of the order takes place. If this sum were to be equal or bigger, there would be no gain in the substitution. Assuming that an order is found which will decrease the longest run by substitution, the procedure is repeated. Another stream might now have the longest run. Let us call this stream "critical". The shortest order of this

stream is taken out and added to the shortest run, if the new sum is smaller than the sum of the critical stream. And so this is carried on.

If no economics can be gained by packing of orders, the algorithm proceeds with exchange of orders. This exchange takes place, if an exchange of orders in two streams will reduce the longest run. The largest order in the critical stream is exchanged with the smallest order in the shortest run. If this meets the criterion - reduction of the longest run - the orders are actually exchanged in the production schedule. If not, the second largest order in the critical will then be tried and then the third and etc. If this fails then the same procedure is repeated with the second smallest order of the shortest run. And so on.

If no exchange attempt is yet succesful the algorithm proceeds as follows: the orders in the next shortest run will now be compared with the orders in the critical streams, following identical rules. If no further economic exchange could be realized after inspecting all the orders in the m-1 streams against all the orders in the critical stream, the algorithm terminates.

If a stream has two rows, a color feasibility check always precedes a packing or an exchange of orders, but otherwise the procedure is identical. (The initial production schedule would have m empty rows - one in each stream - these would all then be the shortest run. The "tie" will be resolved

arbitrarily, i.e. a packing can be done to any of the m empty rows initially).

Before presenting an example, let us briefly summarize the principles of the color-selection algorithm.

1. Assign orders with most frequent colors to stream one, two, ..., m-1 respectively and all remaining orders to the last unoccupied stream,

2. attempt to pack the smallest order of the critical stream into the row having the shortest run,

3. if a packing is made, repeat step2; if packing cannot be performed any more, go to 4,

4. attempt to exchange orders according to previously mentioned rules,

5. if exchange is made repeat step 4; if no exchange can be performed any more routine terminates.

The computer listings of the color-selection algorithm is given in Appendix C.

An Example,

A list of orders might look as follows:

Order No.	Quantity	Color
1	150M	bk, ye
2	75M	bk, or
3	180M	gr, or
4	120M	bk, ye
5	140M	bk, br
6	60M	bl, re
7	100M	bk, --
8	30M	bl, or
9	90M	br, gr
10	70M	re, or

Color code: bk = black, bl = blue, br = brown, gr = green, or = orange, re = red, ye = yellow.

Assuming a m = 3 stream (j = 1, 2, 3) machine set up, the vectors containing all orders will be (bk and or being the first and second most frequent colors respectively):

Order No.	1	2	4	7
Quantity	150M	175M	120M	100M
Color	bk, ye	bk, or	bk, ye	bk, --

Order No.	3	8	10
Quantity	180M	30M	70M
Color	gr, or	bl,or	re, or

Order No.	5	6	9
Quantity	40M	60M	90M
Color	ye, br	bl, re	br, gr

Starting Production Schedule,

	Order No.	1	2	4	7
Stream 1	Quantity	150M	75M	120M	100M
	Color	bk, ye	bk, or	bk, ye	bk, --

	Order No.	3	8	10
Stream 2	Quantity	180M	30M	70M
	Color	gr, or	bl, or	re, or

	Order No.	5	6	9
Stream 3	Quantity	140M	60M	90M
	Color	ye, br	bl, re	br, gr

Since a packing of Order No. 2 into the second stream leads to a reduction of the longest run (Stream 1; 445M), this order will be removed from Stream No. 1 and assigned to Stream No. 2.

With this change the production schedule appears as follows:

	Order No.	1	4	7	
Stream 1	Quantity	150M	120M	100M	
	Color	bk, ye	bk, ye	bk, --	
	Order No.	3	8	10	2
Stream 2	Quantity	180M	30M	70M	75M
	Color	gr, or	bl, or	re, or	bk, or
	Order No.	5	6	9	
Stream 3	Quantity	140M	60M	90M	
	Color	bk, br	bl, re	br, gr	

Since a packing of the shortest order (Order No. 7) of the longest run (Stream 1) to the shortest run (Stream 3), would <u>not</u> result in a reduction of the longest run, an exchange of orders will be tried.

Exchange Order No. 4, Stream 1 with Order No. 6, Stream 3. (Recall the rules: try the largest order of the longest run, then the second largest order etc. as exchange quantity with smallest quantity in shortest run. If no exchange meets the criterion, i.e. the reduction of the longest run, start with the largest quantity in the longest run and try to exchange

with smallest quantity in the second shortest run etc. If all noncritical streams (longest run) were tried and no exchange could be made, go to the second smallest quantity in the noncritical streams etc., beginning again with the largest quantity of the longest run, second largest quantity etc.).

The indicated exchange of orders leads to the following production schedule:

	Order No.	1	6	7	
Stream 1	Quantity	150M	60M	100M	
	Color	bk, ye	bl, re	bk, --	
	Order No.	3	8	10	2
Stream 2	Quantity	180M	30M	70M	75M
	Color	gr, or	bl, or	re, or	bk, or
	Order No.	5	4	9	
Stream 3	Quantity	140M	120M	90M	
	Color	bk, br	bk, ye	br, gr	

Exchange now Order No. 2, Stream 2, with Order No. 6, Stream 1, leaving the following production schedule:

	Order No.	1	2	7	
Stream 1	Quantity	150M	75M	100M	
	Color	bk, ye	bk, or	bk, --	
	Order No.	3	8	10	6
Stream 2	Quantity	180M	30M	70M	60M
	Color	gr, or	bl, or	re, or	bl, re

	Order No.	5	4	9
Stream 3	Quantity	140M	120M	90M
	Color	bk, br	bk, ye	br, gr

We now exchange Order No. 9 with Order No. 2 (third and first Stream respectively), leaving the following <u>final</u> production schedule:

	Order No.	1	9	7	
Stream 1	Quantity	150M	90M	100M	
	Color	bk, ye	br, gr	bk, --	
	Order No.	3	8	10	6
Stream 2	Quantity	180M	30M	70M	60M
	Color	gr, or	bl, or	re, or	bl, re
	Order No.	5	2	4	
Stream 3	Quantity	140M	75M	120M	
	Color	bk, br	bk, or	bk, ye	

Chapter IV

EVALUATION OF THE MATHMATICAL ALGORITHMS AND EXPERIMENTAL RESULTS

Discussion and criticism of the linear programming formulation

As can be seen from the very small example, the linear programming formulation tends to become large. A realistic set of orders in this formulation would involve thousands of variables. This is not surprising since inherently combinational problems have large matrix forms.

The matrix (Table 1) could be reduced if instead of employing the simplex-method as a solution algorithm the dual-simplex method was applied. The latter method does not require artificial and slack variables. To get an idea of the dimension of a problem formulated in this manner, for a realistic set of orders, assume the following:

m = 3 streams

n = 10 orders

k = 12 (the quotient largest ordered quantity/ smallest ordered quantity, rounded to larger integer),

there will be, number of variables

$$X_{ijk} \quad - \quad 3 * 10 * 12 = 360$$

constraint inequalities,

$$n + (m * k) \qquad = 46$$

The index k is diputable because it is highly unlikely that in a feasible solution k will be of this magnitude. This

is a safe assumption, since the increasing cost coefficients insure the tendency to bring into the solution small values of k. In fact, the linear programming formulation is based on the assumption that the optimal solution will contain variables with the smallest possible cost coefficients, but will still accommodate all constraints.

Although there are computer programs[4] which could handle problems with a realistic set of orders in the above manner, the run time to generate a solution would be too long to be economically justified. With this comment we shall discard the linear programming formulation as a practical and economical solution technique to the scheduling problem.

Evaluation of exhaustive search as a solution technique

The formulation of the scheduling problem, as defined in this study, in a linear programming framework has shown that the size of the resulting matrix is very large for any realistic set of orders. It will now be shown that an exhaustive search also presents similar difficulties.

If n = 15 orders are to be scheduled on m = 3 streams, we could arrive at 1,728,000 distinct schedules (employing the formula given on page 14). This number could be reduced somewhat by thoughtful reflection, however, the number still remains quite large and so the method is clearly impractical.

---

[4] GE 600, Revised Simplex Product Form, constraints ≤ 1,000, variables ≤ 6,000.

Evaluation of the suboptimizing algorithms

In this section we describe the testing and costing procedures used to evaluate the largest-quantity-first algorithm and the color-selection algorithm.

The object in testing the algorithms is to determine their merits on an economic ground. We will generate production schedules with different sets of data. Then for each of the schedules its production costs will be computed. Also for each of the tested set of orders an ideal, usually not attainable production schedule will be made up and its value (in monetary terms), will be computed. The quotient ideal cost/actual cost will give us an indication of the merits of the devised algorithms and they will be rated (evaluated) accordingly.

The data for testing the algorithms are simulated. Actual data, representing orders received by a paper box company over some time period, most likely will not be as varied as simulated data. That is why hypothetical data (simulated) seem to be much better suited for the evaluation of the algorithms (the same set of data will be used for both algorithms).

Cost of an ideal schedule

An ideal schedule, as mentioned previously is usually not attainable. It will have no run outs -- this condition imposes non feasibility, e.g. you cannot have the same number of cartons printed in each stream if there are two orders with 10M

and 25M to be processed on a m = 2 stream set up. An ideal schedule puts 17.5M cartons in each stream, rendering it non feasible.

The costs incurred by the different sources, of an ideal schedule, will be reduced to a common denominator as the basis for comparison with the schedules as arrived at by the devised algorithms.

A list will be given below of the various sources of expenditures involved to print paper cartons and their magnitudes:

Cost of paper board

The price of this major raw material is currently Dol. 8.66 CWT[5], as obtained from a paper box company.

We shall confine our discussion to a typical calipher weighing 185 pounds per 1,000 square inch.

To obtain a monetary value the following formula is applicable:

Cost of consumed paper board = (no. of cartons printed in longest run) * (no. of rows) * (conversion factor) * 8.66

where the conversion factor takes into account the thickness of the paper and the width of paper for one stream.

---

[5] CWT = One Hundred Weight -- Standard measurement unit in the paper industry.

Let us consider machine set ups having 2, 3 or 4 streams, each stream having 1 row (for testing purposes).

The value of the conversion factor equals 0.64, i.e. it is the weight of 1M cartons in CWT units for the previously defined calipher and the considered carton size. (Abbreviated CPB).

Printing press cost

The time to print a set of orders is a function of the longest run in a production schedule, and then of the number and kinds of make-ready's, the latter being taken care of in the next item of this list.

A modern printing press is capable of printing 20M impressions per hour per row (333 per minute) while running at normal speed.

The machine cost was estimated by management at Dol. 22.00 per hour at a typical box plant. This estimate might lack precision, or better it might not reflect the true cost, due to the utilized accounting procedure. However as will be seen, it could be off by a factor of two and still make but a small difference in the total production cost for a set of orders. (Abbreviated PPC).

Cost of make-ready's

A make-ready is defined in Appendix A (Glossary): "Time and action taken to make plate changes (and eventually color changes) and do the necessary adjustments when the machine is at standstill".

The time to be allotted for make-ready's is given at the end of this section (Table 2). The accumulated time (in minutes) for make-ready's by computing the cost of a production schedule, will be multiplied by Dol. 0.36 (this is the same as for the machine running time -- the attending crew has to be paid whether controlling the printing press while running or when busy with make-ready's. The power cost difference is negligible). (Abbreviated CMR).

Scrap paper cost

The value of scrap paper is currently Dol. 1.42 CWT. For a definition of scrap paper see the Glossary (Appendix A)

In monetary terms we express as follows:

Scrap paper cost per stream = (number of cartons printed in longest run) - (number of cartons printed in considered stream) * (conversion factor) * 1.42,

where "conversion factor" has same meaning as before. (Abbreviated SPC).

We continue:

Cost of production schedule = CPB + PPC + CMR - SPV

Cost of ideal schedule = CPB + PPC + CMR

An example -- Computing the cost of an ideal Schedule, assuming a set of orders having the following attributes:

Order No.	Quantity	Color(s)
1	20M	blue, green
2	50M	blue, green
3	80M	blue, green
4	30M	blue, green
5	20M	brown
6	50M	brown
7	80M	brown
8	30M	brown
9	20M	red, orange
10	50M	red, orange
11	80M	red, orange
12	30M	red, orange

For a m = 3 stream machine set up, an ideal production schedule would contain:

Order No. 1 to Order No. 4 in Stream 1 (in sequence)

Order No. 5 to Order No. 8 in Stream 2 (in sequence)

Order No. 9 to Order No. 12 in Stream 3 (in sequence)

We continue as follows:

Cost of ideal Schedule = CPB + PPC + CMR,

where all members of the right hand side of the equation have been defined previously.

CPB = (20M + 50M + 80M + 30M) * (3) * (0.64) * 8.66

CPB = 3000.00 Dol.[6]

PPC = (20M + 50M + 80M + 30M) * (1/20) * 22.00

PPC = 198.00 Dol.

---

[6] Slide rule accuracy.

$CMR^7 = (57^8 + 57 + 57 + 57) * 0.36$

CMR = 82.00 Dol.

We now compute the,

Cost of the ideal Schedule = 3000.00 + 198.00 + 82.00

= 3280.00 Dol.

Scope of testing the algorithms

There will be 6 sets of orders, each set containing between 8 and 17 orders. Testing the two algorithms with 6 sets each, will give a clear picture regarding their performance. The number of orders per set (randomly between 8 and 17) were chosen to reflect reality. At least in the box plant visited by the author, this was the typical range of orders per set.

A simulated order will have associated with it:

1. quantity to be scheduled,
2. one or two colors.

As to the quantity of an order it will vary between 10M and 100M. Again these limits were chosen to reflect reality. Colors will be identified by a four-digit number and two letters. The key is as follows:

---

7 The cost for make-ready's will be computed for an actual production schedule and the same value will be used for the computation of the cost of an ideal schedule: this cost factor has nothing but a minor influence on the quotient, ideal cost/actual cost.

8 At each stop, i.e. after printing 20M, 50M, 80M cartons, 10 printing plates have to be replaced and adjusted. For time-factor see Table 2.

black	7200	bk
blue	6700	bl
brown	4500	br
green	8700	gr
orange	9500	or
purple	3800	pu
red	1700	re
red	2900	re
yellow	5500	ye

The simulation will be carried out in a fashion, that on the average 8/10 ths of all orders will have two colors specified as is more or less the case in reality. Naturally there are no orders having the same color specified twice.

Simulating by the Monte Carlo technique,

We will describe now how we arrived at the first set of orders. "A Table of Random Numbers" of reference (6), page 3-84 will be used.

How many orders will be in set 1, 2, ..., 6 ?

Start at the upper left hand corner of the random number table and draw numbers going down column 1. Consider random numbers between 8 and 17 (inclusive). Set one will contain 16 orders.

Quantity associated with order 1, 2, ..., 16 ?

Start at the upper right hand corner of the random number table and go down along column 30, then column 29 and

so on. The last digit of the random number will be rounded to five, if 1, 2, 3 or 4 and rounded to zero if 6, 7, 8 or 9, since the finest breakdown for a quantity will be 5M, to conform to the industrial practice (e.g. for random number 38 read 40, or for the random number 21 read 25). Skip random numbers with the first digit zero. Continue along the same column to draw random numbers for quantities in set 2 etc.

Set 1:	Order No.	1	2	3 cont'd
	Quantity	95M	75M	10M

Colors associated with order 1, 2, ..., 16 ?

Begin at the upper left hand corner of the random number table, read down along column 1, then column 2, and so on. the first digit refers to the color, the second digit of a random number defines whether an order will have one or two colors specified, according to the following rule: 0, 1, 2 for second digit will have one color per order; 3, 4, ..., 9 for second digit will have two colors per order. Skip random numbers with the first digit zero. Continue in the same column to draw random numbers for colors in set 2, 3, ... 6.

An example,

With these basic rules established, the first simulated set of orders will look as follows (the remaining sets were obtained in an identical manner):

Set 1

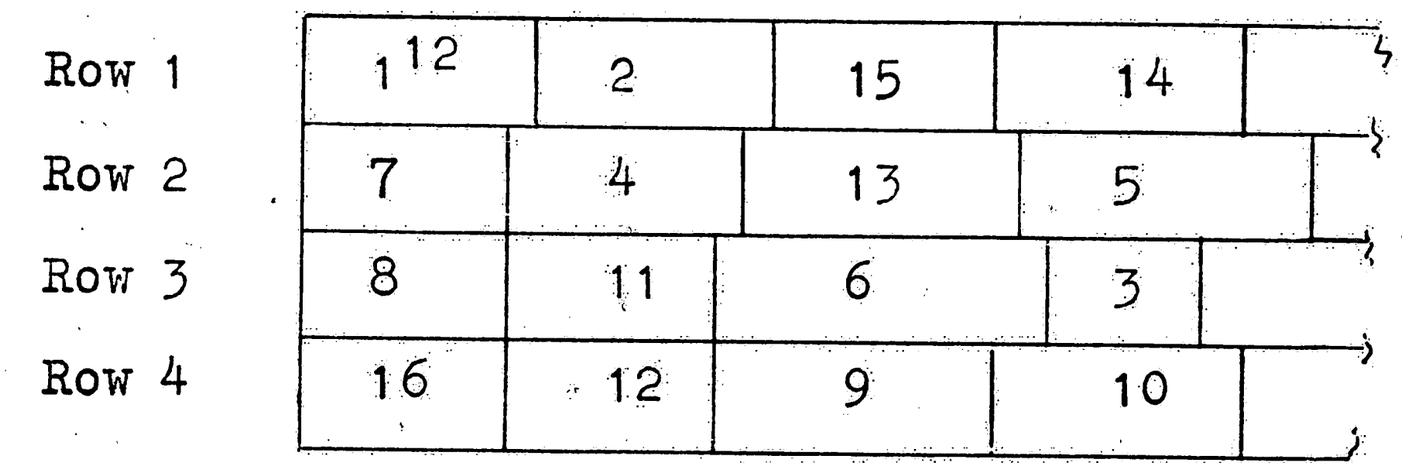
Order No.	1	2	3	4	5	6	7	8
Quantity	95M	75M	10M	80M	30M	65M	80M	80M
Color(s)	95[11] or	17 re	55 ye	87 gr	38 pu	17 re	29 re	38 pu
	17 re	-- --	17 re	67 bl	55 ye	29 re	55 ye	72 bk

Order No.	9	10	11	12	13	14	15	16
Quantity	55M	30M	75M	75M	55M	40M	30M	80M
Colors(s)	55 ye	17 re	67 bl	29 re	17 re	38 pu	17 re	29 re
	95 or	38 pu	72 bk	17 re	-- --	67 bl	67 bl	-- --

Computing the cost of a production schedule,

As a sample we will compute the cost of a production schedule as arrived at by using "Largest-quantity-first algorith (see production schedule Test Order Set No. 1, see Table 3 and 4):

Pictorially we can show as follows,



---

11 The last two digits (zeros) are omitted.

12 Order No.

and continue,

CPB = 245 * 4 * 0.64 * 8.66

= 5440.00 Dol.

PPC = 245 * (1/20) * 22.00

= 270.00 Dol.

CMR = (69[13] + 20 + 62 + 25 + 33 + 60 + 36 + 33) * 0.366

= 124.00 Dol.

SPV = (5 + 15 + 5) * 0.64 * 1.42

= 23.00 Dol.

finally,

Cost of production schedule = 5440.00 + 270.00 + 124.00

- 23.00

= 5811.00 Dol.

Computing the cost of an ideal schedule,

As defined previously, the cost of the ideal schedule becomes (same order set as before, i.e. production schedule Test Order Set No. 1, arrived at by using Largest-quantity-first algorithm, see Table 3 and 4):

CPB = (240 + 245 + 230 + 240) * 0.64 * 8.66

= 5300.00 Dol.

PPC = (240 + 245 + 230 + 240) * (¼) * (1/20) * 22

= 262.00

CMR = 124.00 Dol.

---

13 As per Table 2.

finally,

Cost of ideal schedule = 5300.00 + 262.00 + 124.00

= 5686.00 Dol.

and the efficiency quotient becomes,

$\frac{5686.00}{5811.00}$ = 0.975 which measures the effectiveness of the used scheduling technique.

Brief discussion of the experimental results

The results obtained with simulated data are presented in Table 5. The entries in the column headed "Efficiency quotient" are the results of interest.

The Largest-quantity-first algorithm scheduled Test Order Set No. 2 with an efficiency quotient of 0.906. This is the lowest entry in the column. The production schedule is for a four row printing press set up. Ceteris paribus, the Color-selection algorithm scheduled with an efficiency quotient of 0.985; which is well above the combined average of 0.972. The mean of the Largest-quantity-first algorithm's efficiency quotient is 0.965, the mean of the Color-selection algorithm's efficiency quotient is 0.979.

From the twelve test runs, ten times, the Color-selection algorithm produced schedules with an equal or higher efficiency quotient than the Largest-quantity-first algorithm. A paired-sample test[14] was carried out which showed with 95% probability, a significant difference between the performances of the two

---

[14] The binomial probability of having 9 positive differences in 11 trials (there was one tie) is 0.0269 (See Table I, page 356 of reference (2)).

respective algorithms. This is not surprising, however, it is the structure of the algorithms that brings out these results: the Largest-quantity-first algorithm converges very fast, in fact all assignments to a production schedule are made only once, i.e. they are final. The Color-selection algorithm tries first to avoid color changes in the individual rows -- recall that in this algorithm, orders having the most frequent colors are assigned to the same row. Then a packing and an exchange of orders takes place and convergence is only realized after a lengthy search for fitting orders. The price we pay for the latter algorithm's generally better quality decisions is in computer run time.

The recorded solution time on a GE-225 8K computer is as follows:

Algorithm used	No. of rows	Order-set No.	Computer[17] run time
L-Q-F	3	1 to 6	3.4[18]
C-S-A	3	1 to 6	7.2
L-Q-F	4	1 to 6	3.3
C-S-A	4	1 to 6	9.0

---

17 Solution time in minutes.

18 The times shown are the times required to compile the FORTRAN IV program, generate the solution and print the production schedule.

Table 2

Standard Time for Make Ready's[9]

2 Plates[10]	No Color Change	20 min.
2 "	1 " "	25 "
4 "	No " "	28 "
4 "	1 " "	33 "
4 "	2 " "	36 "
6 "	No " "	38 "
6 "	1 " "	43 "
6 "	2 " "	46 "
6 "	3 " "	49 "
8 "	No " "	51 "
8 "	1 " "	54 "
8 "	2 " "	57 "
8 "	3 " "	60 "
8 "	4 " "	62 "
10 "	No " "	57 "
10 "	1 " "	60 "
10 "	2 " "	62 "
10 "	3 " "	64 "
10 "	4 " "	66 "
10 "	5 " "	68 "
12 "	No " "	60 "
12 "	1 " "	63 "
12 "	2 " "	65 "
12 "	3 " "	67 "
12 "	4 " "	69 "
12 "	5 " "	71 "
12 "	6 " "	73 "

---

9 As per standards of the box plants visited by the author. Based on normal crew, i.e. one machine operator and one helper.

10 Two plates are used for each color printed.

TEST ORDER SET NO. 1

ORDERS SORTED TO QUANTITY

ORDER	1	SET 1	QUA	95.0M	COLORS	9500	OR	1700 RE
ORDER	7	SET 1	QUA	80.0M	COLORS	2900	RE	5500 YE
ORDER	8	SET 1	QUA	80.0M	COLORS	3800	PU	7200 BK
ORDER	16	SET 1	QUA	80.0M	COLORS	2900	RE	0 --
ORDER	4	SET 1	QUA	80.0M	COLORS	8700	GR	6700 BL
ORDER	11	SET 1	QUA	75.0M	COLORS	6700	BL	7200 BK
ORDER	12	SET 1	QUA	75.0M	COLORS	2900	RE	1700 RE
ORDER	2	SET 1	QUA	75.0M	COLORS	1700	RE	0 --
ORDER	6	SET 1	QUA	65.0M	COLORS	1700	RE	2900 RE
ORDER	9	SET 1	QUA	55.0M	COLORS	5500	YE	9500 OR
ORDER	13	SET 1	QUA	55.0M	COLORS	1700	RE	0 --
ORDER	15	SET 1	QUA	40.0M	COLORS	1700	RE	6700 BL
ORDER	14	SET 1	QUA	30.0M	COLORS	3800	PU	6700 BL
ORDER	10	SET 1	QUA	30.0M	COLORS	1700	RE	3800 PU
ORDER	5	SET 1	QUA	30.0M	COLORS	3800	PU	5500 YE
ORDER	3	SET 1	QUA	10.0M	COLORS	5500	YE	1700 RE

Table 3

PRODUCTION SCHEDULE LARGEST-QUANTITY-FIRST-ALGORITHM

TEST ORDER SET NO. 1

SET UP HAS M= 4ROWS

ASSIGNMENTS TO ROW 1

ORDER 1	SET 1	QUANTITY	95.0M	COLORS	9500	OR	1700 RE
ORDER 2	SET 1	QUANTITY	75.0M	COLORS	1700	RE	0 --
ORDER 15	SET 1	QUANTITY	40.0M	COLORS	1700	RE	6700 BL
ORDER 14	SET 1	QUANTITY	30.0M	COLORS	3800	PU	6700 BL

ASSIGNED QUANTITY ROW 1=240.0M

ASSIGNMENTS TO ROW 2

ORDER 7	SET 1	QUANTITY	80.0M	COLORS	2900	RE	5500 YE
ORDER 4	SET 1	QUANTITY	80.0M	COLORS	8700	GR	6700 BL
ORDER 13	SET 1	QUANTITY	55.0M	COLORS	1700	RE	0 --
ORDER 5	SET 1	QUANTITY	30.0M	COLORS	3800	PU	5500 YE

ASSIGNED QUANTITY ROW 2=245.0M

ASSIGNMENTS TO ROW 3

ORDER 8	SET 1	QUANTITY	80.0M	COLORS	3800	PU	7200 BK
ORDER 11	SET 1	QUANTITY	75.0M	COLORS	6700	BL	7200 BK
ORDER 6	SET 1	QUANTITY	65.0M	COLORS	1700	RE	2900 RE
ORDER 3	SET 1	QUANTITY	10.0M	COLORS	5500	YE	1700 RE

ASSIGNED QUANTITY ROW 3=230.0M

ASSIGNMENTS TO ROW 4

ORDER 16	SET 1	QUANTITY	80.0M	COLORS	2900	RE	0 --
ORDER 12	SET 1	QUANTITY	75.0M	COLORS	2900	RE	1700 RE
ORDER 9	SET 1	QUANTITY	55.0M	COLORS	5500	YE	9500 OR
ORDER 10	SET 1	QUANTITY	30.0M	COLORS	1700	RE	3800 PU

ASSIGNED QUANTITY ROW 4=240.0M

Table 4

Table 5

Scheduling Results - Suboptimizing algorithms with simulated data

Set No.	Orders per set	No. of rows	Algorithm used	Cost of ideal S.	Cost of actual S.	Efficiency quotient
1	16	3	L-Q-F[15]	5782	5799	0.996
1	16	3	C-S-A[16]	5807	5991	0.973
2	12	3	L-Q-F	4220	4612	0.915
2	12	3	C-S-A	4231	4456	0.950
3	16	3	L-Q-F	5364	5531	0.970
3	16	3	C-S-A	5364	5364	1.000
4	16	3	L-Q-F	5030	5101	0.984
4	16	3	C-S-A	5029	5029	1.000
5	16	3	L-Q-F	4869	5022	0.970
5	16	3	C-S-A	4868	4932	0.988
6	11	3	L-Q-F	3302	3382	0.977
6	11	3	C-S-A	3295	3449	0.955
1	16	4	L-Q-F	5686	5811	0.975
1	16	4	C-S-A	5688	5813	0.976
2	12	4	L-Q-F	4330	4788	0.906
2	12	4	C-S-A	4330	4391	0.985
3	16	4	L-Q-F	5277	5370	0.980
3	16	4	C-S-A	5277	5364	0.983
4	16	4	L-Q-F	4922	5002	0.981
4	16	4	C-S-A	4940	5020	0.985
5	16	4	L-Q-F	4796	4896	0.980
5	16	4	C-S-A	4796	4896	0.980
6	11	4	L-Q-F	3233	3439	0.940
6	11	4	C-S-A	3244	3337	0.975

[15] Largest-quantity-first algorithm

[16] Color-selection algorithm

Chapter V

CONCLUSIONS AND EXTENSIONS

The subject of this study has concerned decision procedures for use in the lower echolon of the scheduling problem hierarchy. The investigation was confined to a single-machine scheduling problem, i.e. the assignment of orders to a printing press. It has been demonstrated that this problem by itself can be a large scale decision problem; standard linear programming or exhaustive search routines are not practical as methods of solution. Recourse has therefore been made to suboptimizing algorithms.

Some test runs have been performed using these algorithms with simulated data. The resulting production schedules had efficiency quotients, as defined in this study, of about 0.90 to 1.00, with an average of 0.972. The general conclusion can be drawn that feasible computer procedures have been developed for application to production scheduling. The Largest-quantity-first algorithm scheduled with an average coefficient quotient of 0.965, the Color-selection algorithm with an average coefficient quotient of 0.979. This difference is statistically significant at the 95% level.[19]

Although all discussions in this study have been confined to the scheduling of box container printing presses, the

---

[19] Paired-sample test, see page 38 for explanation.

approaches taken are fairly general. Applications might be found in the production of at least some of the following items (5, 9):

1. cellophane,
2. photographic film,
3. plate glass,
4. plastics,
5. production of sheet metal,
6. textiles.

Suggestion for further study

The author has studied the actual practice of printing press scheduling to gain an understanding of the technological and economic nature of the problem. The motivation was then to start with an open mind in devising a different approach to the problem, essentially independent of current practice.

All of the models presented in this paper are static in the sense that the group of orders to be scheduled is taken as fixed, which means that its requirements must be filled exactly. It has been pointed out that actual scheduling in a paper box factory differs significantly from these conditions. One earlier quoted reference (5) reports that some management in the paper making and paper converting industry have successfully adopted a very flexible ordering policy in agreement with its customers. For example,

1. ordered quantities can be largely over or under shipped,

2. customers store goods supplied early or the paper box plant stores goods manufactured to early.

This paper stresses that over or under supply, as well as keeping manufactured goods in inventory, should be carefully scrutinized and use of this remedies should be made only if production schedules can be obtained which are really close to an optimal production schedule. The first suggestion then is:

1. Can the models presented here be modified to incorporate the more dynamic aspects of actual scheduling?

This question is posed as a basis for further research. It is also hoped that a basic framework of scheduling algorithms presented in this study will stimulate research into the applications of these approaches, especially to adapt these models to actual operating conditions in some of the previously enumerated possible areas of applications.

2. It would be of interest to test the presented algorithms empirically with simulated data. The random numbers drawn from different distributions instead of the used Uniform distribution.

The second suggestion could lead to conclusive results regarding the performance of the algorithms confronted with different data, e.g. it is conceivable that for different distributions the performance of the presented algorithms is quite different.

APPENDIX A

This glossary contains definitions or explanations of some of the more frequently used and less familiar terms employed by the container industry men.

stream	The width of a printing press is subdivided, the divisions being called a stream. In each stream up to three distinct colors can be printed.
row	A stream might be partitioned in one or two rows, depending on the size of the box-container to be printed.
compatibility	Only certain combinations of cardboxes having different sizes and/or designs are feasible, meaning that they can be processed simultaneously (assigned to different streams in a production schedule).
color-compatibility	If a stream is comprised of more than one row, colors of the orders assigned to it must be compatible, meaning that the sum of distinct colors across a stream must not be bigger than three.
run out	The procedure of printing in less than the full number of rows is called a run out (paper is fed across the full width of the machine, but a fraction of this printing takes place).
efficiency	The term refers to the utilization of the machine-width in operation.
scrap paper	Paper board of discrete width fed into the machine, but is only partly printed (run out) constitutes wastage (if in only two of three rows printing is done, 1/3 of the fed-in-cardboard is scrap paper).
shortest run	The smallest sum-quantity in a row of a production schedule.

longest run	The largest sum-quantity in a row of a production schedule.
make ready	Time and action taken to make plate changes and color changes and do the necessary adjustments when the machine is at a standstill.

APPENDIX B

Computer listings of the Largest-quantity-first algorithm.

The appendix gives the computer listings of the programmed algorithm. The listings are well documented with comment statements which will aid in the understanding of the program flow.

```
 FORTRAN IV, CD225H6.004,     JAN   1967
$JOB       NOLIST,NOCARDS
$FORTRAN
C
C
C
C
C
C
C       LARGEST-QUANTITY -FIRST ALGORITHM
C
        COMMON JCON(17,3),QUA(18),JCOL(17,2),JKOL(17,2)
        COMMON JASS(5,18),L,IROW,SQUA(5),LISQ2(18),NC
        READ 650,NSAM
  650   FORMAT (I2)
        DO 550 IJK=1,NSAM
        READ 650,N
        K2=N
        READ 650,NC
C
C       ZERO OUT QUA(N+1)
C
        QUA(N+1)=0.0
C
C       READ DATACARD
C
C       SORTED TO QUANTITY
C
        DO 030 I=1,K2
        READ 010,(JCON(I,J),J=1,3),QUA(I),JCOL(I,1),JKOL(I,1),
       1JCOL(I,2),JKOL(I,2)
  010   FORMAT (3A3,F6.0,I4,A2,I4,A2)
  030   CONTINUE
        DO 075 I=1,K2
        LISQ2(I)=I
  075   CONTINUE
        PRINT 005,IJK
  005   FORMAT (1H1,////, 8X,18HTEST ORDER SET NO.,I2,////)
        PRINT 125
  125   FORMAT (1H0,7X,25HORDERS SORTED TO QUANTITY,//)
        DO 031 JJ=1,K2
        I=LISQ2(JJ)
        PRINT 015,(JCON(I,J),J=1,3),IJK,QUA(I),JCOL(I,1),JKOL(I,1),
       1JCOL(I,2),JKOL(I,2)
  015   FORMAT (1H0, 7X,3A3,3HSET,I2,3X,3HQUA,X,F5.1,1HM,3X,6HCOLORS,
       12X,I4,X,A2,3X,I4,X,A2)
  031   CONTINUE
C
        L=1
C
C       THE FIRST NC ASSIGNMENTS ARE MADE FROM LISQ LISTS
C
C       WITHOUT CALLING ON SUBROUTINE MIQUA
C
C       NC = NUMBER OF ROWS READ IN AS PARAMETER
C
        DO 100 J=1,NC
        JASS(J,L)=LISQ2(J)
  100   CONTINUE
        JA = NC
  150   CALL MIQUA
C
```

```
C       INREASE L BY ONE
        L=L+1
        DO 120 J=1,NC
C
C       ZERO OUT ALL L-ASSIGNMENTS
C
        JASS(J,L)=N+1
  120 CONTINUE
        JA=JA+1
C       IROW FROM SUBROUTINE MIQUA
C       IROW IS SMALLEST ROW)
C       ASSIGN NOW TO IROW
C                                                          DEBUG
        JASS(IROW,L)=LISQ2(JA)
C                                                          DEBUG
        IF(JA-K2)150,200,200
C
CC      ROUTINE TERMINATES IF ALL ORDERS OF LIQUA ARE ASSIGNED
CC
CC      BACK TO MAIN PROGRAM
C
  200 PRINT 085,IJK
  085 FORMAT(1H1, 8X,19HPRODUCTION SCHEDULE,2X,
     132HLARGEST-QUANTITY-FIRST-ALGORITHM,2X,///,
     29X, 18HTEST ORDER SET NO.,I2,/)
      PRINT 999,NC
  999 FORMAT (8X,14H SET UP HAS M=,I2,4HROWS,//)
      DO 095 K=1,NC
      SQUA(K)=0.0
      PRINT 077,K
  077 FORMAT(1H0, 8X,18HASSIGNMENTS TO ROW,I2,/)
      DO 060 M=1,L
      I=JASS(K,M)
      IF ((N+1)-I) 060,060,196
  196 PRINT 020,(JCON(I,J),J=1,3),IJK,QUA(I),JCOL(I,1),JKOL(I,1),
     1JCOL(I,2),JKOL(I,2)
  020 FORMAT (    9X,3A3,3HSET,I2,3X,8HQUANTITY,X,F5.1,1HM,3X,6HCOLORS,
     11X,I4,X,A2,3X,I4,X,A2)
      SQUA(K)=SQUA(K)+QUA(I)
  060 CONTINUE
      PRINT 025,K,SQUA(K)
  025 FORMAT(///,8X,22H ASSIGNED QUANTITY ROW,I2,1H=,F5.1,1HM,//)
  095 CONTINUE
  550 CONTINUE
      CALL EXIT
      END
```

```
$FORTRAN
      SUBROUTINE MIQUA
      COMMON JCON(17,3),QUA(18),JCOL(17,2),JKOL(17,2)
      COMMON JASS(5,18),L,IROW,SQUA(5),LISQ2(18),NC
C
C
C     SUBROUTINE SELCTS SMALLEST ACCUMULATED QUANTITY
C
C
C     IN ANY OF THE NC ROWS
      DO 100 I=1,NC
      SQUA(I)=0.0
  100 CONTINUE
      DO 200 K=1,L
      DO 200 J=1,NC
      KO=JASS(J,K)
      SQUA(J)=SQUA(J) * QUA(KO)
  200 CONTINUE
      BIG=99999.9
      DO 110 J=1,NC
      IF(BIG - SQUA(J))110,110,120
  120 IROW = J
      BIG = SQUA (J)
C
  110 CONTINUE
C
C     SMALLEST ROW QUANTITY CALLED IROW WILL BE  TRANSFERRED
      RETURN
      END
```

APPENDIX C

Computer listings of the Color-selection algorithm. The appendix gives the computer listings of the programmed algorithm. The listings are well documented with comment statements which will aid in the understanding of the program flow.

```
 FORTRAN IV, CD225H6.004,    JAN   1967
$JOB       NOLIST,NOCARDS
$FORTRAN
C
C
C
C
C     COLOR-SELECTION ALGORITHM
C
C
C
      COMMON N,L,K2,IMA,LAUS,IROW,IJK,JUMP,NC
      COMMON MFC(5,10),NMFC(5),JASS(6,19),SQUA(6),SUR(6)
      COMMON JCOL(18,2),LISQ2(18),LICO(18),LISO(18),JCL(18,2),JA(18)
      COMMON JCON(18,3),QUA(19),JKOL(18,2)
      COMMON SMAL,BIG,SWITCH
C
C     THERE WILL BE SIX SETS OF ORDERS
C
      DO 600 LMN=1,6
C
      JDUM=LMN
C
C     MAIN PROGRAM CALLS ON
C
C     DIFFERENT SUBROUTINES TO
C     PERFORM ALL TASKS NECESSARY
C
C
  100 CALL ORDIN
C
      SWITCH=0.0
C
      JUMP=0
C
      CALL PPRIN
C
      CALL COSEL
C
  200 CALL SUQUA
C
      CALL ROQUA
C
      CALL QUARO
C
      CALL SUQUA
C
      IF(JUMP)300,300,400
  300 CALL PACK
C
      GO TO 200
C
  400 CALL EXCHA
C
      IF(SWITCH)200,200,500
C
  500 CALL PRINT
C
  600 CONTINUE
C
  700 CALL EXIT
      END
```

```
$FORTRAN
      SUBROUTINE ORDIN
C
C
C     SUBROUTINE READS ORDERS OF
C
C     ONE  TEST SET
C
      COMMON N,L,K2,IMA,LAUS,IROW,IJK,JUMP,NC
      COMMON MFC(5,10),NMFC(5),JASS(6,19),SQUA(6),SUR(6)
      COMMON JCOL(18,2),LISQ2(18),LICO(18),LISO(18),JCL(18,2),JA(18)
      COMMON JCON(18,3),QUA(19),JKOL(18,2)
      COMMON SMAL,BIG,SWITCH
C
C     N EQUALS NUMBER OF ORDERS PER SET
C
      READ 650,N
  650 FORMAT (I2)
C
C     NC EQUALS NUMBER OF ROWS IN
C
C     PRINTING PRESS SET UP
C
      READ 650,NC
C
C     IJK EQUALS NUMBER OF TEST SET
C
      READ 650,IJK
      DO 030 I=1,N
      READ 010,(JCON(I,J),J=1,3),QUA(I),JCOL(I,1),JKOL(I,1),
     1JCOL(I,2),JKOL(I,2)
  010 FORMAT (3A3,F6.0,I4,A2,I4,A2)
  030 CONTINUE
      QUA(N+1)=0.0
      L=N
      K2=N
      RETURN
      END
```

```
$FORTRAN
      SUBROUTINE PPRIN
C
CC
C
CC    SUBROUTINE PRINTS OUT ALL
C
CC    ORDERS IN TEST SET
C
CC    ORDERS ARE IN RANDOM SEQUENCE
C
C
      COMMON N,L,K2,IMA,LAUS,IROW,IJK,JUMP,NC
      COMMON MFC[5,10],NMFC[5],JASS[6,19],SQUA[6],SUR[6]
      COMMON JCOL[18,2],LISQ2[18],LICO[18],LISO[18],JCL[18,2],JA[18]
      COMMON JCON[18,3],QUA[19],JKOL[18,2]
      COMMON SMAL,BIG,SWITCH
      PRINT 005,IJK
  005 FORMAT [1H1,////, 8X,18HTEST ORDER SET NO.,I2,////]
      DO 031 I=1,N
      PRINT 015,[JCON[I,J],J=1,3],IJK,QUA[I],JCOL[I,1],JKOL[I,1],
     1JCOL[I,2],JKOL[I,2]
  015 FORMAT [1H0, 7X,3A3,3HSET,I2,3X,3HQUA,X,F5.1,1HM,3X,6HCOLORS,
     12X,I4,X,A2,3X,I4,X,A2]
  031 CONTINUE
      RETURN
      END
```

```
$FORTRAN
      SUBROUTINE COSEL
C
C
C     SUBROUTINE SELECTS ORDERS WITH
C
C      MOST FREQUENT COLORS
C
      COMMON N,L,K2,IMA,LAUS,IROW,IJK,JUMP,NC
      COMMON MFC(5,10),NMFC(5),JASS(6,19),SQUA(6),SUR(6)
      COMMON JCOL(18,2),LISQ2(18),LICO(18),LISO(18),JCL(18,2),JA(18)
      COMMON JCON(18,3),QUA(19),JKOL(18,2)
      COMMON SMAL,BIG,SWITCH
C
      DO 055 II=1,K2
      LISQ2(II)=II
C                                                    DEBUG
  055 CONTINUE
C     ZERO OUT LICO
C
C     LICO STORES INDICES OF COLORS (FIRST INDEX OF BOTH COLORS)
C
      DO 110 II=1,9
      LICO(II)=0
  110 CONTINUE
C
C     RUN THROUGH A LIST WITH A PARTICULAR CONTAINER CAPACITY
C
C     DIVIDE COLOR NUMBER BY THOUSAND AFTER CONVERSION
C
C     TO FLOATING POINT MODE
C
C     THE SECOND AND THIRD DIGIT OF THE
C
C     COLOR NUMBER WILL BE TRUNCATED
C
      DO 100 II=1,K2
C
      I=LISQ2(II)
      DO 100 JJ=1,2
      XJCO=JCOL(I,JJ)
C
C     IF  NO COLOR IS SPECIFIED GO TO NEXT ITEM IN THE LIST
       IF(XJCO) 100,100,120
C
  120 JCO=XJCO/1000.0
      LICO(JCO)=LICO(JCO)+1
  100 CONTINUE
C
C     LIST **LICO** STORES FIRST DIGIT OF COLOR NUMBER
C
C     NC EQUALS NJMBER OF ROWS IN WHICH
C
C     ASSIGNMENTS NEED TO BE MADE
C
C     NC READ IN AS PARAMETER
C
C
C     WE ARE INTERESTED IN THE
```

```
C
C       (NC-1)=NCC MOST FREQUENT COLORS
C
C       DO LOOP SELECTS THE LARGEST  **LICO**
        NCC=NC-1
        DO 300 II=1,NCC
        JSMA=-5
        DO 250 I=1,9
        IF(JSMA-LICO(I))200,250,250
  200   JCO=I
C
C       INDEX OF CURRENTLY MOST FREQUENT **LCO LIST** WILL BE PRESERVED
C
        JSMA=LICO(I)
  250   CONTINUE
C       ZERO OUT THE LARGEST SELECTED **LICO**
C
C
C       START A NEW LIST **LISO** WITH NCC MOST FREQUENT COLORS
        LICO(JCO)=0
        LISO(II)=JCO
  300   CONTINUE
C
C       DO LOOP SELECTS THE LARGESST **LICO**
C       SET COLOR NUMBERS EQUAL A DUMMY
C
C       THIS ALLOWS TO ZERO OUT ASSIGNED COLORS
        DO 350 JJ=1,K2
        II=LISO2(JJ)
        JCL(II,1)=JCOL(II,1)
        JCL(II,2)=JCOL(II,2)
  350   CONTINUE
C
C       DO LOOP COMPARES MOST FREQUENT
C
C       COLORS OF **LISO**
C
C       WITH STORED COLOR NUMBER
C
C       IF REAL COLOR NUMBER IN THE
C
C       ONE THOUSAND RANGE AGREES WITH MOST
C
C       FREQUENT COLOR IN **LISO** ASSIGNMENT CAN BE MADE
C
        DO 097 JJ=1,NCC
        NMFC(JJ)=0
  097   CONTINUE
        DO 500 I=1,NCC
C
C       MULTIPLY LIST CODE
C
C       NMFC(I) RECORDS NUMBER OF
C
C       THE MOST FREQUENT COLORS
C
C
        III=0
        JES = LISO(I)*1000
        DO 500 KK=1,K2
```

```
      II=LISQ2(KK)
      DO 500 JJ=1,2
      IF(JCL(II,JJ))500,500,550
  550 IF(JES-JCL(II,JJ)) 400,450,500
  400 IF((JES+1000)-JCL( II,JJ))500,500,450
C
CCC   MFC IS LIST OF MOST FREQUENT COLORS
CCC
CCC   FIRST SUBSCRIPT FOR COLOR, SECOND COUNTS ORDERS OF PART, COLOR
CCC
C
  450 III=III+1
      NMFC(I)=III
      MFC(I,III)=II
      LISQ2(KK)=N+1
      DO 501 JJ=1,2
      JCL(II,JJ)=0
  501 CONTINUE
  500 CONTINUE
C
CCC   DO LOOP STS ALL POSSIBLE ASSIGNMENTS TO N+1
CCC
CCC   QUA(N+1) IS SET TO ZERO
CCC
CCC   SUBROUTINE ASSIGNS TO THE LAST **NC** ROW ALL ORDERS LEFT
C
      DO 217 M=1,NC
      DO 217 J=1,K2
      JASS(M,J)=N+1
  217 CONTINUE
C
CC    DO LOOP ASSIGNS TO NCC ROWS THE ORDERS
C
      DO 218 J=1,NCC
      KI=NMFC(J)
      DO 218 M=1,KI
      JASS(J,M)=MFC(J,M)
  218 CONTINUE
      LK=0
      DO 367 KK=1,K2
      II=LISQ2(KK)
      IF (II-(N+1))975,367,367
  975 I=JCL(II,1)
      IF(I) 367,368,366
  368 I=JCL(II,2)
      IF(I) 367,367,366
  366 LK=LK+1
      JASS(NC,LK)=LISQ2(KK)
  367 CONTINUE
      RETURN
      END
```

```
$FORTRAN
      SUBROUTINE PRINT
C
CC
C
C
CCC     SUBROUTINE PRINTS FINAL PRODUCTION SCHEDULE
C       WITH PROPER HEADINGS
CC
C

      COMMON N,L,K2,IMA,LAUS,IROW,IJK,JUMP,NC
      COMMON MFC[5,10],NMFC[5],JASS[6,19],SQUA[6],SUR[6]
      COMMON JCOL[18,2],LISQ2[18],LICO[18],LISO[18],JCL[18,2],JA[18]
      COMMON JCON[18,3],QUA[19],JKOL[18,2]
      COMMON SMAL,BIG,SWITCH
  200 PRINT 085,IJK
  085 FORMAT[1H1, 8X,19HPRODUCTION SCHEDULE,2X,
     125HCOLOR-SELECTION-ALGORITHM,2X,///,
     29X, 18HTEST ORDER SET NO.,I2,/]
      PRINT 999,NC
  999 FORMAT [8X,14H SET UP HAS M=,I2,4HROWS,//]
      DO 095 K=1,NC
      SQUA[K]=0.0
      PRINT 077,K
  077 FORMAT[1H0, 8X,18HASSIGNMENTS TO ROW,I2,/]
      DO 060 M=1,N
      I=JASS[K,M]
      IF [[N+1)-I] 060,060,196
  196 PRINT 020,[JCON[I,J],J=1,3],IJK,QUA[I],JCOL[I,1],JKOL[I,1],
     1JCOL[I,2],JKOL[I,2]
  020 FORMAT [   9X,3A3,3HSET,I2,3X,8HQUANTITY,X,F5.1,1HM,3X,6HCOLORS,
     11X,I4,X,A2,3X,I4,X,A2]
      SQUA[K]=SQUA[K]+QUA[I]
  060 CONTINUE
      PRINT 025,K,SQUA[K]
  025 FORMAT[///,8X,22H ASSIGNED QUANTITY ROW,I2,1H=,F5.1,1HM,//]
  095 CONTINUE
      RETURN
      END
```

```
$FORTRAN
      SUBROUTINE SUQUA
C
CC
C
C
C     SUBROUTINE COMPUTES AGGRAVATED SUM  (SUR( ) ) IN EACH ROW
C
      COMMON N,L,K2,IMA,LAUS,IROW,IJK,JUMP,NC
      COMMON MFC(5,10),NMFC(5),JASS(6,19),SQUA(6),SUR(6)
      COMMON JCOL(18,2),LISQ2(18),LICO(18),LISO(18),JCL(18,2),JA(18)
      COMMON JCON(18,3),QUA(19),JKOL(18,2)
      COMMON SMAL,BIG,SWITCH
C
CC    SUR ( ) WILL BE STORED IN COMMON
C
      DO 010 J=1,NC
      SUR(J)=0.0
      DO 010 K=1,L
      I=JASS(J,K)
      SUR(J)=SUR(J)+QUA(I)
  010 CONTINUE
      RETURN
      END
```

```
$FORTRAN
      SUBROUTINE ROQUA
C
CC
CCC
CC
C
C
      SUBROUTINE SORTS ALL ROWS ACCORDING TO QUANTITY
C
C
      COMMON N,L,K2,IMA,LAUS,IROW,IJK,JUMP,NC
      COMMON MFC[5,10],NMFC[5],JASS[6,19],SQUA[6],SUR[6]
      COMMON JCOL[18,2],LISQ2[18],LICO[18],LISO[18],JCL[18,2],JA[18]
      COMMON JCON[18,3],QUA[19],JKOL[18,2]
      COMMON SMAL,BIG,SWITCH
C
CC
C     LARGEST QUANTITY IN ROW 1, SMALLEST QUANTITY IN ROW NC
C
      NCC=NC-1
      DO 140 K=1,NCC
      AQU=SUR[K]
      LL=K+1
      DO 130 J=LL,NC
      IF[AQU-SUR[J]]120,130,130
  120 DO 150 KK=1,L
      JSTO=JASS[J,KK]
      JASS[J,KK]=JASS[K,KK]
  150 CONTINUE
      AQU=SUR[J]
C     PRINT 160,AQU,J
  160 FORMAT [F6.1,I2]
  130 CONTINUE
      DO 010 JJ=1,NC
      SUR[JJ]=0.0
      DO 010 KK=1,L
      I=JASS[JJ,KK]
      SUR[JJ]=SUR[JJ]+QUA[I]
  010 CONTINUE
  140 CONTINUE
      RETURN
      END
```

```
$FORTRAN
      SUBROUTINE QUARO
C
CC
C
C     SUBROUTINE SORTS ALL NC ROWS INDIVIDUALLY TO QUANTITY
C
      COMMON N,L,K2,IMA,LAUS,IROW,IJK,JUMP,NC
      COMMON MFC[5,10],NMFC[5],JASS[6,19],SQUA[6],SUR[6]
      COMMON JCOL[18,2],LISQ2[18],LICO[18],LISO[18],JCL[18,2],JA[18]
      COMMON JCON[18,3],QUA[19],JKOL[18,2]
      COMMON SMAL,BIG,SWITCH
C
C     LARGEST QUANTITY WILL BE IN FIRST POSITION [L=1], SECOND LARGEST ETC.
C
      DO 140 J=1,NC
      LL=1
      LIK=L-1
      DO 140 K=1,LIK
      M=JASS[J,K]
      AQU=QUA[M]
      LL=LL+1
      DO 130 KK=LL,L
      NN=JASS[J,KK]
      IF[AQU-QUA[NN]]120,120,130
  120 AQU=QUA[NN]
      JSTO=JASS[J,K]
      JASS[J,K]=JASS[J,KK]
      JASS[J,KK]=JSTO
  130 CONTINUE
C     PRINT 160, AQU,QUA[NN]
  160 FORMAT[2F6.1]
  140 CONTINUE
      RETURN
      END
```

```
$FORTRAN
      SUBROUTINE PACK
C
CC
CC
C     SUBROUTINE TRIES TO PACK ORDERS
C
      COMMON N,L,K2,IMA,LAUS,IROW,IJK,JUMP,NC
      COMMON MFC(5,10),NMFC(5),JASS(6,19),SQUA(6),SUR(6)
      COMMON JCOL(18,2),LISQ2(18),LICO(18),LISO(18),JCL(18,2),JA(18)
      COMMON JCON(18,3),QUA(19),JKOL(18,2)
      COMMON SMAL,BIG,SWITCH
C
CC    LOGIC IS BASED ON THE FACT, THAT ALL ORDERS IN ALL ROWS
CC
CC    HAVE BEEN PREVIOUSLY SORTED, WITHIN ROWS AND ALSO SEQ. OF ROWS
C
      LT=L+1
      NK=NC+1
      NKK=NC-1
      DO 150 LM=1,NKK
      KL=NK-LM
      DO 150 IJ=1,L
      JK=LT-IJ
      NN=JASS(KL,JK)
      IF(NN-N+1)150,160,160
  160 DO 130 K=1,L
      M=JASS(1,K)
      IF(M-N+1) 140,130,130
  140 IF(SUR(KL)+QUA(M)-SUR(1))100,130,130
  100 JASS(KL,JK)=JASS(1,K)
      JASS(1,K)=N+1
      GO TO 300
  130 CONTINUE
  150 CONTINUE
      JUMP =1
  300 RETURN
      END
```

```
$FORTRAN
      SUBROUTINE EXCHA
C
CC
C
CC
C     SUBROUTINE TRIES TO EXCHANGE ORDERS
C
      COMMON N,L,K2,IMA,LAUS,IROW,IJK,JUMP,NC
      COMMON MFC[5,10],NMFC[5],JASS[6,19],SQUA[6],SUR[6]
      COMMON JCOL[18,2],LIS02[18],LICO[18],LISO[18],JCL[18,2],JA[18]
      COMMON JCON[18,3],QUA[19],JKOL[18,2]
      COMMON SMAL,BIG,SWITCH
C
CCC   LOGIC IS BASED ON THE FACT, THAT ALL ORDERS IN ALL ROWS
C
CCC   HAVE BEEN PREVIOUSLY SORTED, WITHIN ROWS AND ALSO SEQ. OF ROWS
C
      LT=L+1
      NK=NC+1
      NKK=NC-1
      DO 501 LM=1,NKK
      KL=NK-LM
      DO 150 IJ=1,L
      JK=LT-IJ
      NN=JASS[KL,JK]
      IF[NN-N+1]160,150,150
  160 DO 130 K=1,L
      M=JASS[1,K]
      IF[M-N+1] 140,130,130
  140 IF[SUR[KL]+QUA[M]-QUA[NN]-SUR[1]]100,130,130
  100 IF [QUA[M]-QUA[NN]]130,130,110
  110 JSTO=JASS[1,K]
      JASS[1,K]=JASS[KL,JK]
      JASS[KL,JK]=JSTO
      GO TO 300
  130 CONTINUE
  150 CONTINUE
  501 CONTINUE
      SWITCH=1.0
  300 RETURN
      END
```

BIBLIOGRAPHY

1. Biegel, John E., "OR in the Job Shop", Preprint AD 67-180 by the ASTME, 1967.

2. Freund, John E., "Mathematical Statistics", Prentice-Hall, Inc., Englewood Cliffs, N.J., 1962.

3. Gomory, R.E., "The Trim Problem", IBM Systems Journal, September 1962, pp. 77-82

4. Holstein, William K., "Mathematical Models for Paper Mill Scheduling", Ph. D. Dissertation 1964, Purdue University.

5. Hox, Guenther, "Erweiterte Anwendung von Linear Programming bei der Belegung von Papiermaschinen", Ablauf und Planungsforschung, Vol. 5, No. 1, pp. 26-33, 1964.

6. Maynard, H.B., "Industrial Engineering Handbook", Second Edition, McGraw-Hill Book Co., Inc.,1963.

7. Morgan, James I., "Survey of Operations Research", Paper Mill News, March 13, 1961, pp. 10-11.

8. Pierce, John F., "Some Large-Scale Production Scheduling Problems in the Paper Industry", Prentice-Hall, Inc., 1964.

9. Reith, P.F., "Computers en beperking van snijafval", SIGMA, Vol. 8, No. 3, pp. 61-64, 1962.

10. Tonge, Fred M., "A Heuristic Program for Assembly Line Balancing", Prentice-Hall, Englewood Cliffs, N.J., 1961.

11. Van Wormer, Theodore, A., "The Trimmer - A Heuristic Solution to the Trim Problem in the Corrugated Container Industry", Ph.D. Dissertation 1963, Carnegie Institute of Technology.

VITA

The author was born on March 27, 1940 in Willingen, Hesse, Germany. His parents are Wilhelm and Herta Saure. He is a German citizen.

He was brought up in Willingen and attended the primary and the high schools (mathematisch-naturwissenschaftliches Gymnasium) there. He received the degree of Ingenieur des Maschinenbaus from Staatliche Ingenieurschule Saarbruecken, Saarland (Germany) in July 1964.

From November 1964 he was employed by the Allis-Chalmer Ltd., Montreal, Province of Quebec, Canada. He was engaged with the construction of components of rotary air-compressors, design of lubrication systems and electro-mechanical control systems.

He entered the Graduate School of Lehigh University, Bethlehem, Pennsylvania, in February 1966. He was a Teaching Assistant during the academic year 1966/67 in the Department of Industrial Engineering.